User Guide for Network Visualization Tool

The tool has been developed in R using the “shiny” package. In addition to shiny, we require the following packages installed:

1. igraph: to plot the networks, calculate network centrality measures
   * to install igraph type “*install.packages(‘igraph’)*” in the R command prompt
2. rcolorbrewer: to color the nodes in the network (in the code, this is done on the basis of industry; this can be changed)
   * to install rcolorbrewer, type “*install.packages(‘RColorBrewer’)*” in the R command prompt
3. shinysky: provides an autocomplete textbox, which is not included in shiny
   * to install shinysky, you need to first install a library called devtools by typing “*install.packages(‘devtools’)*”. Once devtools in installed type in the following command “*devtools::install\_github(‘AnalytixWare/ShinySky’)*”

## **Inputs**

1. graph\_edgelist.csv

This file contains the edges of the network in a single column – odd indexes contain the starting node, even nodes contain the terminating node. This is important because the network is a directed one.

**companyNames = read.csv("graph\_edgelist.csv", stringsAsFactors = FALSE, header = FALSE)**

1. **Industry\_Final.csv**

**The tool colors the nodes based on the industry to which the company belongs. This file contains the industry information – column 1 contains the company names and column 2 contains the corresponding industry.**

**industryNames = read.csv("Industry\_Final.csv", stringsAsFactors = FALSE, header = FALSE)**

1. **betweenness.csv**

**Betweenness centrality is an important network measure that tells how connected a node is in the network. The betweenness algorithm is computationally intensive and takes a large amount of time to run, therefore I first get the betweenness scores of all the nodes and store it in a file called betweenness.csv – column 1 contains the node name and column 2 contains the betweenness scores.**

**To generate betweenness scores, run the following code at R prompt (please note that the algorithm will take a few minutes to run)**

**g = graph(edges = companyNames$V1, directed = TRUE)**

**g = simplify(g, remove.multiple = TRUE, remove.loops = TRUE)**

**b = betweenness(g, v = V(g))**

## **User Interface**

**The code is written in two parts (this is how all shiny apps are structured): i) UI elements; ii) backend code that defines the logic (called “server”) that handles all the interactions**

**The UI code is again divided into two parts:**

* **a sidebar panel: this contains all the input widgets i.e. user preferences can be set using these elements. each widget is assigned a unique id (this is used by the backend code)**
  + **for a list of all available input widgets see** [*http://shiny.rstudio.com/gallery/widget-gallery.html*](http://shiny.rstudio.com/gallery/widget-gallery.html)
* **a main panel: we can display the output in the main panel. Similar to the sidebar, each element here is also assigned a unique id. Shiny works on the principle of “reactive” programming i.e. as soon as any input is changed, the code re-runs and displays the new output in the main panel.**
  + **for a list of all available output widgets see “*https://shiny.rstudio.com/tutorial/***

***lesson4/*”**

**The “render” function: the reactive ability of shiny is derived from the render function. For each of the output widget defined in the main panel, we can use a render function (for e.g. renderPlot, renderText, etc.). all the code in the server block must come inside the render function.**

## **igraph functions used**

* **make\_ego\_graph:** This function finds the vertices not farther than a given limit from another fixed vertex, these are called the neighborhood of the vertex.
  + **for syntax type “help(‘make\_ego\_graph’)” at the R command prompt**
* **plot.igraph: to plot the network; igraph provides several layouts to plot a network (this is one of the inputs provided in the sidebar). by layout we mean the x-y coordinates of each node on the plot. we use these x-y coordinates to identify the node that is clicked by the user.**